VEGA corporation

LOWYAの信頼性向上とNew Relic

NRUG SRE支部 Vol.1

2022-05-13

株式会社ベガコーポレーション 小原一真 @kazumax55

本日のテーマ

“俺たちのSREとNew Relic”

今日話すこと

- ベガコーポレーションのSRE

- やってよかった取り組み（AWS）

NewRelicの活用事例と併せてご紹介します！

自己紹介

@kazumax55

小原一真（Kazuma Kohara）

VEGA corporation

技術戦略部 SREグループ グループ長

🏬東京支社（本社は福岡）

趣味

🏕キャンプ 🎣釣り 🧖サウナ

好きなNew Relicの機能

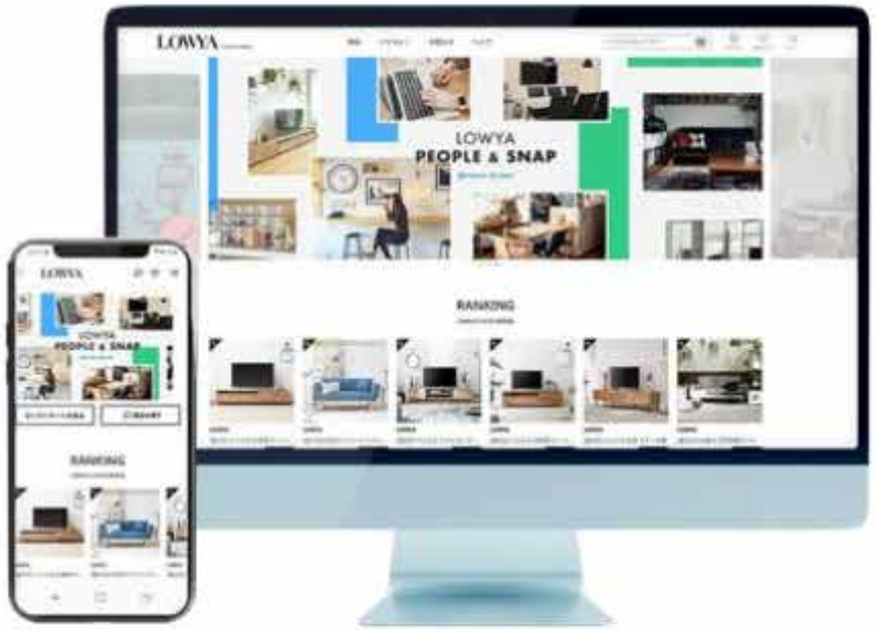
APMのCompare with last week

会社紹介

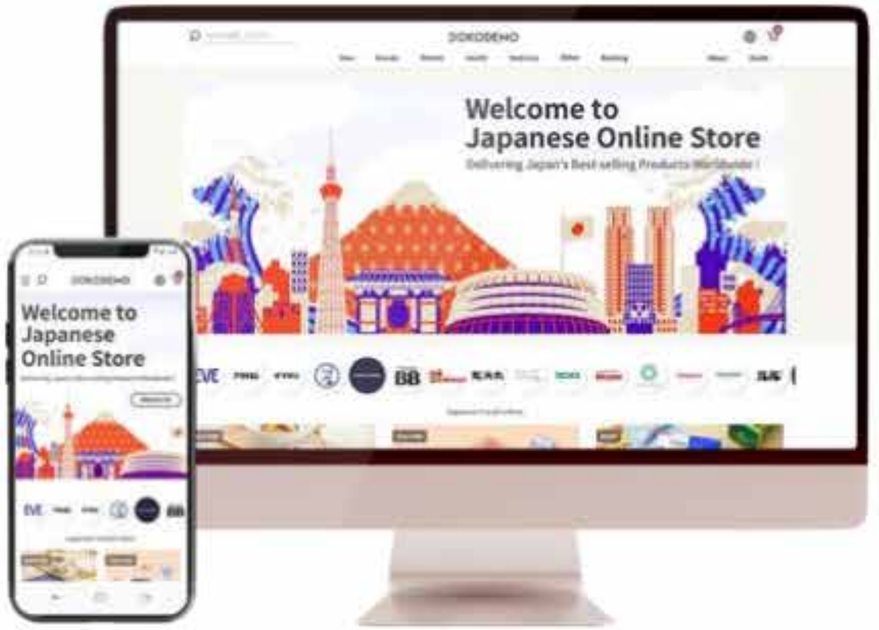
2つのEコマース事業

“家具・インテリアEC事業のLOWYA”を基軸として営み、さらに海外ユーザーをターゲットとした
“越境ECプラットフォーム事業のDOKODEMO”を展開しております。

LOWYA
TREND IS HERE.



DOKODEMO



LOWYA

TREND IS HERE.

LOWYA AR

LOWYA 360

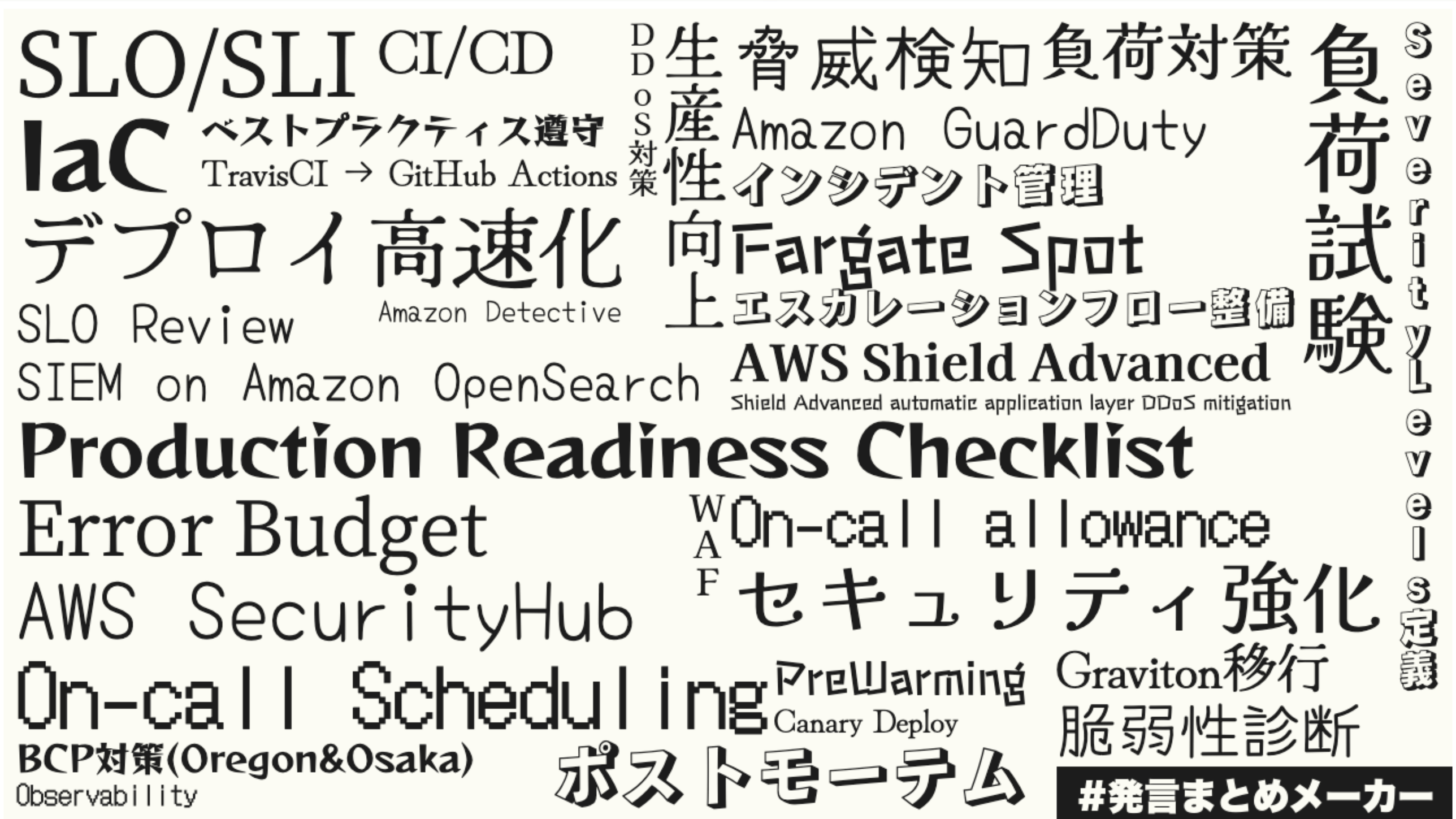
特徴：長いバリューチェーン

ベガコーポレーションのSRE

ベガコーポレーションのSRE

- 設立から4年ちょっと(2022/05現在)
- メンバー数 : 4名
- ミッション : **”業務拡大に合わせ高い信頼性を維持/提供できること”**
  - 事業側の目標に対して、システムがボトルネックにならない状態を維持提供し続ける
  - 課題が浮上しても仕組で解決し、品質を高める事ができる
- Team Topologies : Enabling team
  - 特定の開発チームには属さず、全社横断で信頼性向上に取り組みつつ、SREの文化/プラクティスを浸透させている

やってよかった取り組み


SLO/SLI
CI/CD
IaC
ベストプラクティス遵守
TravisCI → GitHub Actions
デプロイ高速化
SLO Review
SIEM on Amazon OpenSearch
DDoS対策
生産性向上
Amazon Detective
脅威検知
Amazon GuardDuty
負荷対策
インシデント管理
Fargate Spot
エスカレーションフロー整備
AWS Shield Advanced
Shield Advanced automatic application layer DDoS mitigation
負荷試験
Severity levels定義
Production Readiness Checklist
Error Budget
AWS SecurityHub
WAF
On-call allowance
セキュリティ強化
On-call Scheduling
PreWarming
Canary Deploy
Graviton移行
脆弱性診断
BCP対策(Oregon&Osaka)
Observability
ポストモーテム
#発言まとめメーカー


SLO/SLI
CI/CD
IaC
ベストプラクティス遵守
TravisCI → GitHub Actions
デプロイ高速化
DDoS対策
生産性向上
脅威検知
負荷対策
Amazon GuardDuty
インシデント管理
Fargate Spot
エスカレーションフロー整備
負荷試験
Severity levels定義
Amazon Detective
SLO Review
SIEM on Amazon OpenSearch
AWS Shield Advanced
Shield Advanced automatic application layer DDoS mitigation
Production Readiness Checklist
Error Budget
WAF
On-call allowance
AWS SecurityHub
セキュリティ強化
On-call Scheduling
PreWarming
Canary Deploy
Graviton移行
脆弱性診断
BCP対策(Oregon&Osaka)
ポストモーテム
Observability
#発言まとめメーカー

セキュリティ強化

- Observability for security -

セキュリティ強化

- 重要視している3つのポイント
  - セキュリティ基準の遵守
  - 脅威検知
  - インシデントレスポンス

セキュリティ基準の遵守

- なぜ重要？
  - システムもセキュリティもナマモノ
    - 日々更新される、セキュリティ基準も随時アップデートされる
      - いつ穴が開くかわからず、人の力では全てをチェックしきれない
  - 脆弱性診断やってるから平気？
    - 診断直後はOKでも…変更しますよね？
- 解決するためには？
  - Observabilityが必要＝**“AWS Security Hub”**

AWS Security Hubとは？

- AWS上の危険な設定/ベストプラクティスに反する設定を検知/管理する機能
  - イベントドリブンに違反を検知し、通知が可能
    - リスク判定付き(Critical, High …)
      - Critialの例:DB/S3 がPublicで公開された
      - Highの例:SGで80/443以外のポートがAnyで公開された
    - 検知理由と修復手順も提示してくれる

AWS Security Hub 運用

- Observability
  - 週イチでスコアを集計してSlackに通知
    - New Relicにも送ってダッシュボードで可視化
- 運用フロー
  - Slack通知をトリガーに見直しを実施
  - 各プロダクトチームはスコア80点以上を維持する
  - ハイリスクなものから優先して対応
    - Critical検知時は他のタスクを止めて最優先で対応
    - 対応が難しいものはSREに気軽に相談

脅威検知

- なぜ重要？
  - サービスには日々膨大な量のイベントが発生している
  - どこで何が起こっているか/いつ標的にされるかわからない
    - 人の力では全てをチェックしきれない
    - 危険/怪しい 動きがあったら検知したい
- 解決するためには？
  - Observabilityが必要＝**”Amazon GuardDuty”**

Amazon GuardDutyとは？

- AWS上の脅威検知サービス
  - あらゆるログを自動収集し、機械学習や脅威インテリジェンスを利用して悪意のある/不審なアクティビティを検知する
    - 例：怪しいAPI呼び出しが多発している, 不正にデータ取得されていそう ...etc
- 運用フロー
  - 検知したらSlack/PagerDutyに通知
    - 重要度Highの場合、SEV:1インシデントとして最高のPriorityで対応する

インシデントレスポンス

- なぜ重要？
  - 問題が発覚したとき、素早く調査を行い原因を特定する為
- 解決するためには？
  - SIEM(Security Information and Event Management)が必要
  - **“Amazon Detective”と“SIEM on Amazon OpenSearch”**
- 運用フロー
  - 何か問題があったときの調査に利用
    - セキュリティインシデント/運用ミス 等

セキュリティ強化：まとめ

- **セキュリティにもObservabilityを導入しよう**
  - セキュリティ基準の準拠：AWS SecurityHub
  - 脅威検知：Amazon GuardDuty
- **何かあったときに備えてSIEMを有効化しよう**
  - Amazon DetectiveとSIEM on Amazon OpenSearch
- **その他 おすすめ**
  - DDoS対策：AWS WAF Sheild Advance & Automatic application layer DDoS mitigation
  - コスト異常検知：AWS Cost Anomaly Detection
  - 脆弱性調査：Amazon Inspector

負荷対策

負荷対策

- LOWYAの負荷対策
  - LOWYAの歴史
  - 過去の対策と結果
    - 第一の壁：LOWYAの日
    - 第二の壁：TV放映
  - 今後の対策

LOWYAの歴史

- 2004/04 〜
  - 主にモール(楽天/Amazon/PayPay)を主体にして事業展開
- 2017/09 LOWYA旗艦店サービス開始
  - ECパッケージベンダーにカスタマイズを依頼して自社ECを構築/運用
- 2018/03
  - LOWYA旗艦店リプレイスプロジェクト開始（自社でフルスクラッチ）
- 2020/08
  - リプレイス前の負荷試験とボトルネック特定でNew RelicのPoCを開始
  - LOWYA旗艦店リプレイス（Ruby on Rails + Vue.js : GraphQL）

第一の壁：LOWYAの日

- 2021/06/08 LOWYAの日
  - 旗艦店ECリプレース後初の大規模セールイベント（全品20%OFF）
- 事前準備
  - 結構強めの規模でインフラ増強
    - Auroraのスケールアップ
    - 自前Pre-Warming(APIタスクの時限式スケールアウト
- 結果は？？？

LOWYAの日の結果

- 日商 5.5億 (前年対比約2倍の売上)を達成！
  - 諸々問題発生したものの、NewRelicの絶大なトラブルシューティング効果で最初の30分で収束！
- 詳細はデブサミ2021夏で話しました！
- 気になる方はQRコードをチェック！

Developers Summit

LOWYA旗艦店における事業成長と
サービス品質向上の両立を目指すSREの挑戦

デブサミ動画

デブサミ記事

LOWYAの日で発生した問題と対策

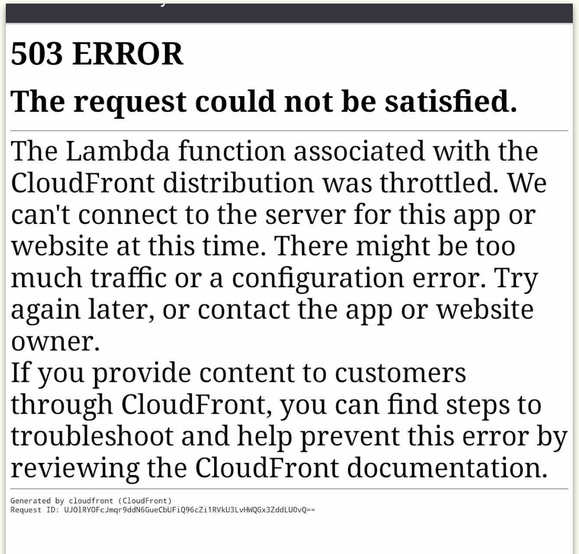
- 問題1:Auroraコネクション枯渇
  - APIの大幅なスケールアウトによって発生
    - → max_connection の値を修正 LEAST({DBInstanceClassMemory/9531392},5000) のLEAST関数を外した
- 問題2:メール配送遅延
  - 注文完了メールの配送能力を上回る速度で注文が入った為、メール配送遅延が発生
    - → メール配送workerのスケールアウトで解決
- 問題3:多重注文
  - 注文完了メール遅延の影響で、一部のお客様が多重注文してしまった → 問題2と3の解消に伴い解決
- 問題4:カード決済閾値
  - 注文が殺到し、秒間決済数の閾値を超えたというエラーが発生 → 決済会社に秒間決済数の上限緩和を依頼

第二の壁：TV放映

- 2021/08/11 今夜くらべてみました放映
  - 家具好きのふなっしーがTV番組内でLOWYA愛を語った
- 事前準備
  - LOWYAの日の倍の規模にインフラ増強
  - AWSへALBのPre-Warming申請
- 結果は？？？

TV放映の結果

- LOWYAの日の倍以上の同時アクセスが発生
  - Lambda@Edgeの閾値を超え、一時的に503エラーを出してしまった
    - AWSに上限緩和して貰う必要がある
    - 負荷が落ち着くまで何もすることができず、悔しい結果となった
    - 以後、社内では **“今くら砲/ふなっしー砲”** などと呼ばれ、恐れられている



今くら砲/ふなっしー砲で発生した問題と対策

- 問題1:Lambda@Edgeの上限問題
  - 画像リサイズやセキュリティ系ヘッダの付与で利用していたL@Eの閾値を超えてしまった
    - → AWSに上限緩和申請を依頼
      - 1 秒あたりのリクエスト 10,000 -> **100,000**
      - 同時実行数 1,000 -> **10,000**
      - **アラート設定を追加(LambdaのThrottle等)**

今くら砲/ふなっしー砲で発生した問題と対策

- 問題2:**実は普段も起きていた事が発覚**
  - 画像つきPush配信時、Throttleが発生していた事が判明
    - Push配信は、送ったタイミングで画像に一斉アクセスが発生するため、L@Eの起動が追いつかずコールドスタートが発生=Throttleが起きた
  - L@Eのヘッダ判定処理を**CloudFront Functions**に切り出す事で恒久対応
    - L@EではProvisioned Concurrencyが設定不可
    - **CF2**は制約も多いが、Pre-Warming不要で秒間数百万アクセスにも耐える事ができる用になった

今後の負荷対策

- 何か簡単かつ飛躍的に改善する手はないか？？
  - 同じ内容のリクエストを毎回処理したくない
  - できればあまり実装は変えたくない
- せや！CDNをかまして同じ内容のリクエストはキャッシュに応答させよう！
  - **GraphQL Caching**だ！

GrahQL Caching

- **Good**
  - コスト削減
    - 毎回ページを表示する度に呼び出されていたものが **キャッシュ期間分の1回** に減る
    - リクエスト数の減少 → APIタスクのスケールイン → コスト削減
  - レイテンシ削減
    - ネットワーク的に近いエッジからオリジンに処理させることなくレスポンスが返せる(**数百msec -> 数十msec**)
  - キャッシュ管理が楽
    - CDN側でTTL, Invalidationが一元管理できる

GrahQL Caching

- **Bad**
  - GraphQLは全てのリクエストがPOSTかつ1つのパスに対して来る為、**キャッシュし辛い**
  - パスが1つだから**キャッシュの部分的な削除**ができない
  - **暴露対策**に注意が必要
  - **情報が少ない**：Akamai, Cloudflare, GraphCDN, Fastly
    - CloudflareでGraphQLリクエストをキャッシュして30msで返すようにした話
      - Cloudflare Workers(Lambda@Edge的なもの)+KVSで実現したという事例

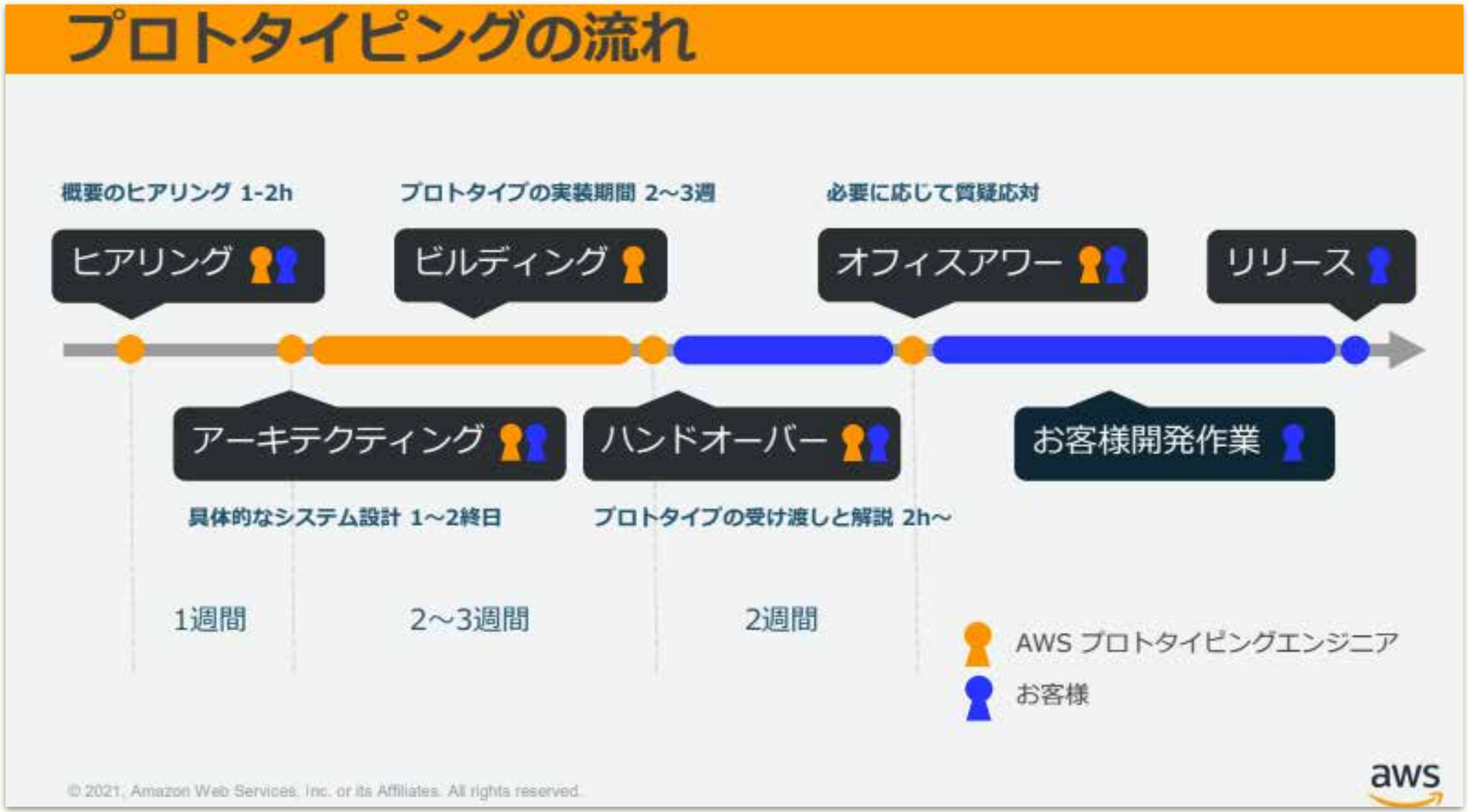
- **Good > Bad**
  - 致命的Badポイントは無し。実現に向け本格的な検討を開始

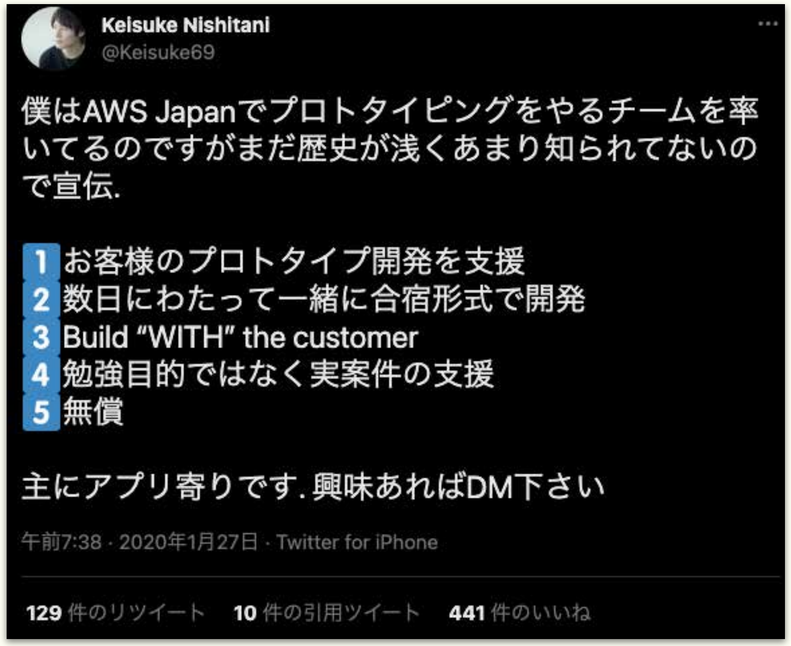
GraphQL Caching

- AWS 担当SAさんにも相談
  - 前々からCloudFrontでGraphQLをキャッシュしたいと相談していた
  - re:Invent2021でもCloudFront GraphQL対応の発表はなかった
  - 他のCDNで実現しようと思っていると正直に話した
    - セキュリティ等も全部ひっくるめて移行検討をすすめていると
  - AWSのソリューションで実現できないか検討させて下さいとの事
    - 数時間後、**“プロトタイピングプログラムはご存知ですか？"**と

プロトタイピングプログラムとは？

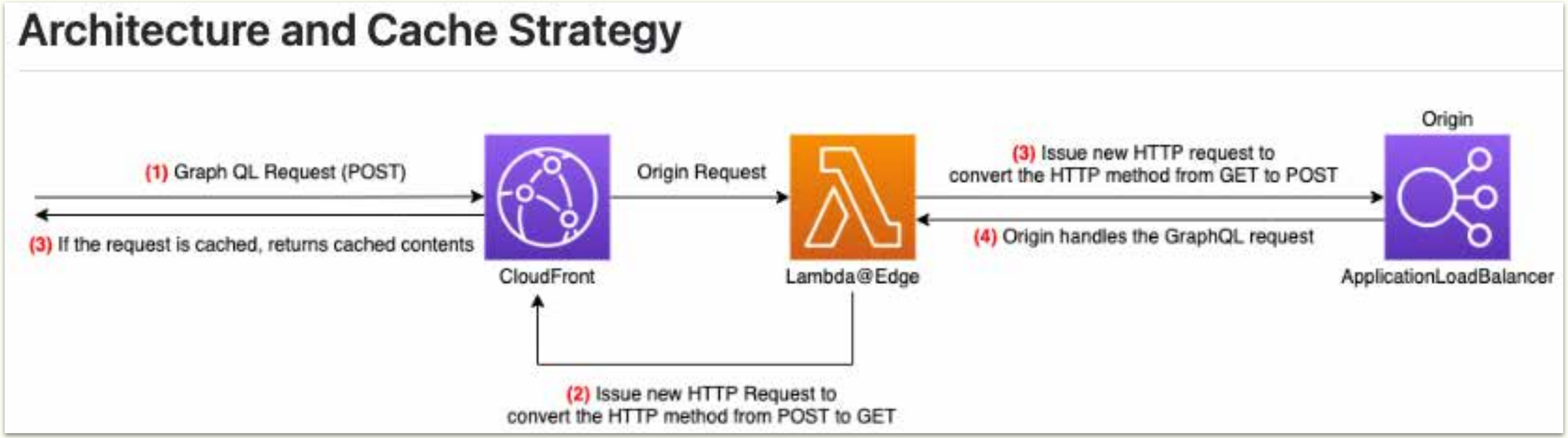
- AWSがユーザの代わりにプロトタイプを開発し、提供してくれる特別なプログラム（**なんと無償で！**）





- 是非お願いします！と即答
- それからスピーディに話をすすめ無事に納品
- AWSがEdgeスペシャリストSAとも相談し、設計した**GraphQL Caching**のアーキテクチャとは？？

GrahQL Caching



- GitHubで公開されており、どなたでも利用可能です
  - aws-samples / amazon-cloudfront-cache-graphql
  - 現在、次回LOWYAの日までの導入に向け絶賛テスト中！

負荷対策：まとめ

- **意外なところに盲点がある**
  - 普段から負荷を気にしている箇所はあまり問題ない
  - そうじゃない所に地雷がある（閾値等）
- **Observerbilityが重要**
  - 何かあった時にすぐ原因特定できる状態にしておく
- **APIキャッシュを検討**
  - 誰がみても同じ応答を返すAPIはキャッシュに応答させる
  - きっと大きな効果を生むはず

SLO Review

SLO Review

- SLOの見直し
- New Relicで測定

SLOの決め方、見直し方？？

- まずSLOを決めて、徐々に見直す
  - 例えば、該当システムのゴールデンシグナルをNewRelicで拾う
    - 平均API応答時間 x 秒以内, ページ表示速度 y 秒以内, エラー率 z %以内, 稼働率 … etc
  - SLOをどうやって見直せば良いかわからなくなる
    - 全ての平均値だと、重要な指標/エラーが埋もれてしまう
    - そもそも全ての平均で見ることに意味はあるのか？
- ではどうすれば良いか？
  - **CUJ(Critical User Journey)**の観点で根本から見直す！

CUJ観点でのSLO Review

- CUJ(Critical User Journey)とは？
  - そのシステムで特に重要視するユーザ体験のこと
- まず第一にCUJを決める
  - そのシステムのユーザは誰か？そのユーザにとって重要なユーザ体験は何か？
    - 例：ECの場合(商品検索、カート、購入など)
- CUJの観点でSLOを考える
  - CUJがエラーも遅延もなくスムーズに使えること
  - ユーザをイラッとさせない秒数は何秒か？とか
    - これをSLOにすべき！

SLO運用フロー

1. CUJを決める(Dev&SRE)
2. SLOを決める(Dev&SRE)
   - Devチーム主体で考えて決めてもらう
     - SLO=SREが決めた面倒なやつと思われないよう、SREはサポートに徹する
   - SREはObserverbilityの提供に責任をもちつつ、SLO等SRE文化の浸透をサポート
     - Observerbilityの導入と、SLO関連ドキュメントの整備やファシリテーション
3. 事業側と合意を取る(Dev)
   - Devチームvs事業 で交渉して、バジェットを勝ち取ってもらう
4. 定期的に見直す(SRE&Dev 🔁 繰り返し)
   - 直近のアラートやメトリクス等の情報を元に定期的な見直しを行う

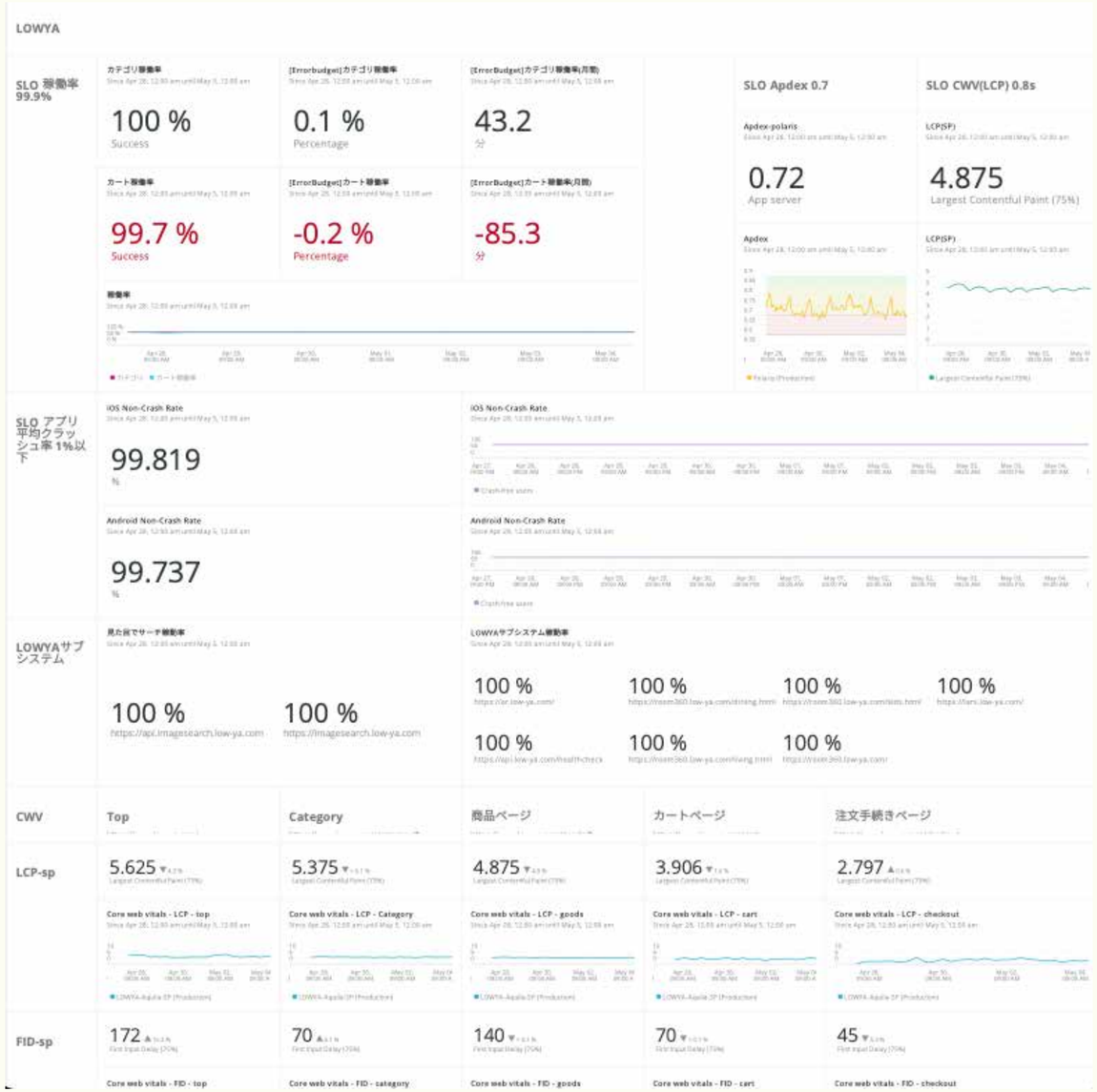
New RelicによるCUJ観点のSLO測定

- Backend
  - APMで対象のCUJで実行されるTransactionを特定し、Apdex/レイテンシ/エラー率 等を見る
    - REST API : Key Transaction / GraphQL : NRQLで絞る
- Frontend
  - BrowserでCUJで呼び出されるパスのCWVやエラー率 等を見る
- Back/Front共通
  - **CUJに影響しないエラーを無視**する(重要)
    - Triage > Erros Inbox や TransactionErrorイベント をみて判断（例：ユーザ起因のクレカ決済エラー等）
- 稼働率
  - SyntheticsでCUJの一連の動きをscriptで定義
    - レイテンシやステータスを見る
- Infrastructure
  - あまり気にする必要はない（SLOに影響がなければOK

これらのSLIを
Alert/SLOとして設定する

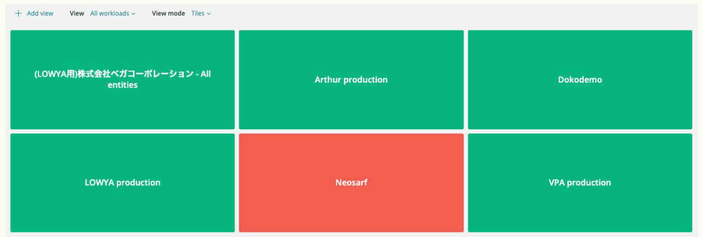
New Relic ダッシュボードの改善

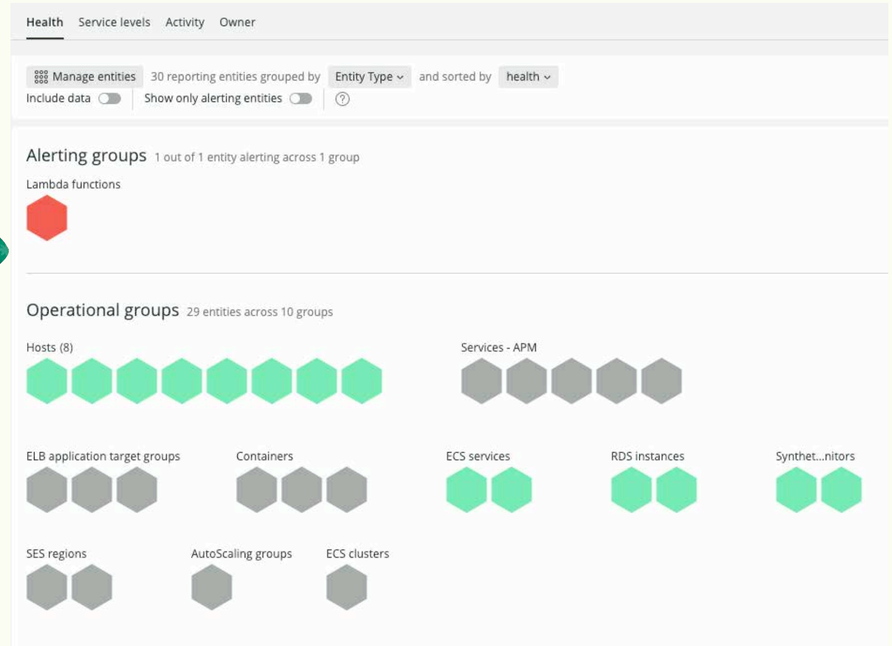
- 今まで
  - 全サービスのSLOをまとめたSLOサマリーダッシュボードを手作り
    - D2Cなので社内サービスが多く表示が重い
    - メンテナンスコストがかかる
- これから
  - 手作りしない
  - Workload views と Service Level Managementを使う

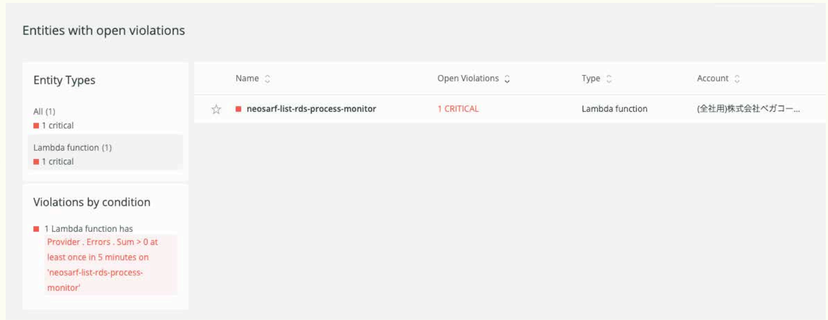


New Relic 新ダッシュボード

- Workloard views はいいぞ！
  - Create Workload > 関連する entities を全部ブチ込む！ > 終わり
  - 勝手にいい感じにダッシュボード化してくれる
    - 足りない部分だけ個別にダッシュボードを作ってWorkloadに追加すればリンクしてくれる
    - トラブルシューティングもスムーズ
  - SLOもService Level Managementを使って再定義
    - 軽い！見やすい！







SLO Review：まとめ

- **CUJ観点でSLO Reviewを実施**
  - 先にCUJを決め、CUJを元に考えていく
  - CUJに関連しないエラーは無視
- **SREがSLOを決めてしまわない**
  - SLOを他人事ではなく自分事として取り組んでもらう為
    - Devチームに決めてもらい、SREは計測と決定をサポートする
- **Workload viewsはいいぞ**
  - 簡単なので是非お試しください
- **今後 New Relicを使ってやりたいこと：ビジネスダッシュボードの整備**
  - ファネル分析/ヒストグラム 等を使ってSLO Reviewに役立てたい（目標を何秒にするか等？
  - 売上/注文数（エラーバジェットを停止時間じゃなくて損失額にしたい


VEGA corporation